Fast Codevector Search Algorithm for 3-D Vector Quantized Codebook

H. B. Kekre, and Tanuja K. Sarode

Abstract—This paper presents a very simple and efficient algorithm for codebook search, which reduces a great deal of computation as compared to the full codebook search. The algorithm is based on sorting and centroid technique for search. The results show the effectiveness of the proposed algorithm in terms of computational complexity. In this paper we also introduce a new performance parameter named as Average fractional change in pixel value as we feel that it gives better understanding of the closeness of the image since it is related to the perception. This new performance parameter takes into consideration the average fractional change in each pixel value.

Keywords—Vector Quantization, Data Compression, Encoding, Searching.

I. INTRODUCTION

Vector quantization (VQ) [1]-[3] is an efficient technique for data compression and has been successfully used in various applications involving VQ-based encoding and VQ-based recognition. The response time is very important factor for real time application [1]. Many type of VQ, such as classified VQ [37], [38], address VQ[37], [39], finite state VQ[37], [40], side match VQ[37], [41], mean-removed classified VQ[37], [42], and predictive classified VQ[37], [43], have been used for various purpose. VQ has been applied to some other applications, such as index compression [37], [44], and inverse half toning [37], [45], [46]. VQ has been very popular in a variety of research fields such as speech recognition and face detection [13], [47], pattern recognition [50]. VQ is also used in real time applications such as real time video-based event detection [13], [48] and anomaly intrusion detection systems [13], [49].

VQ can be defined as a mapping function that maps k-dimensional vector space to a finite set \( \text{CB} = \{C_1, C_2, C_3, \ldots, C_N\} \). The set CB is called codebook consisting of N number of codevectors and each codevector \( C_i = \{c_{i1}, c_{i2}, \ldots, c_{ik}\} \) is of dimension k. The key to VQ is the good codebook.

Codebook can be generated in spatial domain by clustering algorithms or using transform domain techniques [6]-[8]. The method most commonly used to generate codebook is the Linde-Buzo-Gray (LBG) algorithm [3], [4] which is also called as Generalized Lloyd Algorithm (GLA).

In Encoding phase image is divided into non overlapping blocks and each block then converted to the training vector \( X_i = (x_{i1}, x_{i2}, \ldots, x_{ik}) \). The codebook is then searched for the nearest codevector \( C_{min} \) by computing squared Euclidean distance as presented in equation (1) with vector \( X_i \) with all the codevectors of the codebook \( \text{CB} \). This method is called exhaustive search (ES).

\[
\text{min}_{1 \leq j \leq N} d(X_i, C_j)
\]

Where \( d(X_i, C_j) = \sum_{p=1}^{k} (x_{ip} - c_{jp})^2 \) (1)

Although the Exhaustive Search (ES) method gives the optimal result at the end, it involves heavy computational complexity. If we observe the above equation (1) to obtain one nearest codevector for a training vector requires N Euclidean distance computation where N is the size of the codebook. So for M image training vectors, will require M*N number of Euclidean distances computations. It is obvious that if the codebook size is increased to reduce the distortion the searching time will also increase.

In order to reduce the searching time there are various search algorithms available in literature. So far, Partial Distortion search (PDS) [5], equal-average nearest neighbor search (ENNS) [9], the equal average equal variance nearest neighbor search (EENNS) [10], nearest neighbor search algorithm based on orthonormal transform (OTNNS) [11], Partial Distortion Elimination (PDE) [25], triangular inequality elimination (TIE) [36-38], mean distance ordered partial codebook search (MPSS) algorithm [20] ,double test algorithm (DTA) [22], fast codebook search algorithm based on the Cauchy-Schwarz inequality (CSI) [30], fast codebook search based on subvector technique (SRT) [31], the image encoding based on L_2-norm pyramid of codewords [32] and the fast algorithms using the modified L_2-norm pyramid (MLP) [33], fast codeword search algorithm based on VPS+TIE+PDE proposed by Yu-Chen, Bing-Hwang and Chih-Chiang (YBC) in 2008 [34], Eigen vector method (EVM) [21], and others [15], [19], [20], [22] are classified as partial search methods. Some of the partial techniques use data structure to organize the codebook for example tree-based [13], [14], [17], [18], [23], [35] and projection based structure [13], [16], [24]. All these algorithms reduce the

H. B. Kekre is Senior Professor working with Computer Engineering Department, Mukesh Patel School of Technology Management and Engineering, NMIMS University, Vile Parle(w), Mumbai-56, IN. (phone: 9323557897; e-mail: hbkkekre@yahoo.com).

Tanuja K. Sarode, is Ph.D. Scholar from Mukesh Patel School of Technology Management and Engineering, NMIMS University, Vile Parle(w), Mumbai-56, India and Assistant Professor working with Thadomal Shahani Engg. College, Bandra(W), Mumbai-50. (e-mail: tanuja_0123@yahoo.com).
computational cost needed for VQ encoding keeping the
image quality equivalent to Exhaustive search algorithm.

In this paper we propose codebook search algorithm which
uses sorting and centroid technique. The paper also compares
the proposed algorithm with PDS, ENNS, EENNS, and
OTNNS with respect to the execution time and the search
efficiency in the form of ratio evaluated by how many times
the Euclidean distance computation is averagely performed
compared to the size of the codebook. A smaller ratio is better.
We have also introduced a new performance parameter namely
Average Fractional Change in Pixel Value (AFCPV) which is close to human perception. Smaller value of AFCPV
refers to better performance.

In the next section we present some existing search
algorithms. In section III proposed method is given followed
by results in section IV and finally conclusions in section V.

II. REVIEW OF DIFFERENT SEARCH ALGORITHMS

Some existing codevector search algorithms such as PDS,
ENNS, EENNS, and OTNNS are reviewed in this section.

A. Partial Distortion Search (PDS)[5],[36]

The Partial distortion search (PDS) algorithm allows early
termination of the distortion computation between input
training vector and codevector by introducing a premature exit
criterion in the search algorithm. Let \( d_{min} \) be the smallest
distortion obtained so far. If the codevector \( C_i \) satisfies the
condition

\[
\sum_{j=1}^{q}(x_{pj} - c_{ij})^2 \geq d_{min}
\]

Where \( X_p \) is the image training vector and \( j \leq q \leq k \) this
guarantees that \( d(X_p, C_i) \geq d_{min} \).

B. Equal average nearest neighbor search algorithm
(ENNS)[9],[36]

The ENNS algorithm uses the fact that mean of the nearest
codevector is usually close to the mean of the input vector. Let \( m_p \) and \( m_i \) be the mean values of training vector \( X_p \) and
codevector \( C_i \) respectively. If the mean of the codevector \( C_i \) satisfies

\[
m_i \geq m_p + \frac{d_{min}}{k} \quad \text{or} \quad m_i \leq m_p - \frac{d_{min}}{k}
\]

then \( C_i \) will not be the nearest codevector to \( X_p \). To perform
ENNS algorithm mean of all the codevectors should be computed off-line first and stored.

C. Equal Average Variance Nearest Neighbor Search
(EENNS)[9],[36]

EENNS algorithm introduces another significant feature of
vector, the deviation, to reject codevectors. Let \( v_p \) and \( v_i \) are the deviations of \( X_p \) and \( C_i \) respectively, then

\[
(v_p - v_i)^2 \leq d(X_p, C_i)
\]

If the deviation of the codevector \( C_i \) satisfies

\[
v_i \geq v_p + \frac{d_{min}}{k} \quad \text{or} \quad v_i \leq v_p - \frac{d_{min}}{k}
\]

then \( C_i \) will not be the nearest codevector to \( X_p \).

EENNS algorithm performs in two steps. In the first step,
if \( m_i \geq m_p + \frac{d_{min}}{k} \) or \( m_i \leq m_p - \frac{d_{min}}{k} \) then
codevector \( C_i \) can be rejected. Otherwise, in the second,
if

\[
v_i \geq v_p + \frac{d_{min}}{k} \quad \text{or} \quad v_i \leq v_p - \frac{d_{min}}{k}
\]

then codevector \( C_i \) can also be rejected. Hence this algorithms
requires \( N \) mean values and \( N \) deviations of all codevectors.

D. Nearest Neighbor search algorithm based on
orthonormal transform (OTNNS)[11]

Here orthonormal base vectors \( V=(v_1, v_2, ..., v_k) \) for the
Euclidean vector space \( R^k \) are considered. For any \( k-
dimensional vector \( x=(x_1, x_2, ..., x_k) \) it can be transformed to
another Euclidean space defined by the \( k \) orthonormal base
vectors, i.e. \( x = \sum_{j=1}^{k} X_j v_j \) where \( X(X_1, X_2, ..., X_k) \) is the
coefficient vector in the transformed space.

In this algorithm each input vector is a 3-D residual vector and
the orthonormal base vectors are \( v_1 = (1/\sqrt{3},1/\sqrt{3},1/\sqrt{3}) \),
\( v_2 = (1/\sqrt{6},1/\sqrt{6},-2/\sqrt{6}) \), \( v_3 = (1/\sqrt{2},-1/\sqrt{2},0) \)

The conditions for judging possible nearest codevectors are
\[
X_{min} \leq Y_j \leq X_{max} \quad \text{for} \quad 1 \leq i \leq 3
\]

Where \( Y_j = (Y_{j1}, Y_{j2}, Y_{j3}) \) is the coefficient vector of \( C_j \) in the
transformed space, and

\[
X_{min} = X_i - d_{min}
\]
\[
X_{max} = X_i + d_{min}
\]

Preprocessing:

Transform each codevector of the codebook into the space
with base vector \( V=(v_1, v_2, v_3) \) and then sort codevectors in
ascending order with respect to the first elements, i.e. the
coefficients along the base vector \( v_1 \).

Online step:

For searching each input vector \( X_i \) is transformed to obtained
\( \hat{X}_i \). The probable nearby codevector \( Y_j \) is gussed based on
the minimum first element difference criterion. \( d_{min} \) \( X_{min} \)
\( X_{max} \) are calculated. For each codevector \( Y_j \) first check if (6)
is satisfied. If not then \( Y_j \) is rejected else \( d(\hat{X}_i, Y_j) \) is
calculated. If \( d(\hat{X}_i, Y_j) < d_{min} \) then the current closest
codevector to \( \hat{X}_i \) is taken as \( Y_j \) with \( d_{min} \) set to \( d(\hat{X}_i, Y_j) \) and
\( X_{min} \) and \( X_{max} \) are updated accordingly. The procedure is
repeated until best match is found.

III. PROPOSED ALGORITHM

Let \( CB \) be the codebook consisting of \( k \)-dimensional
vectors. In this paper \( k=3 \) consisting of R, G, B component
values of every pixel. First sort the codebook with respect to
the first element of the codevector and then compute the
centroid \( C_0 \) of the first elements of all the codevectors. The
codebook is then divided into two parts based on the centroid,
of the first element, the upper part consists of element values less than this centroid. The upper part of the codebook is again sorted with respect to the second elements of the codevectors and again centroid $c_{00}$ is computed for the second element for the upper part. The process is repeated for the lower part too i.e. lower part of codebook is also sorted with respect to the second elements of the codevectors corresponding the lower part of the codebook and centroid $c_{01}$ is computed for the lower part. Based on the centroid the upper part of the codebook is further divided in to two parts and the above process is repeated, similarly lower part of the codebook is also divided based on to centroid and above process is repeated. For codebook of size $N$ the above process is repeated for $r=(\log_2 N - 3)$ times so get $2^r$ parts of the codebook. The formation of the codebook into subparts is a preprocessing step for encoding is depicted in Fig. 1 as shown below.

Online process:
In Encoding step the first element $x_{i1}$ of image training vector $X_i$ is compared with the $c_0$, if $x_{i1} < c_0$ then $x_{i2}$ is compared with $c_{00}$ else $x_{i2}$ is compared with $c_{01}$ and so on. Once the training vector reaches the last level of tree the nearest codevector is searched form the group of codevectors using Euclidean distance computation. Instead of full search we are dividing codebook into subparts, nearest subpart for the training vector is found out and then closest codevector is searched using exhaustive search applied only to the subpart that is obtained. To locate the nearest subpart $(\log_2 N - 3)$ comparisons are required.

IV. RESULTS
The proposed algorithm is compared with PDS, ENNS, EENNS and OTNNS experimented on 3D mesh model obtained from famous Princeton 3D mesh library [12], Stanford Bunny and Stanford Dragon. The algorithms are implemented using Pentium IV 1.7 GHz 512 MB RAM, using Matlab 6. The algorithm requires $2^r-1$ extra memory space to store the centroids.

Here we have introduced a new performance parameter which is named as Average Fractional Change in Pixel Value (AFCPV) and it is computed as follows:

$$\sum_{MN} \frac{f(x,y) - \hat{f}(x,y)}{f(x,y)}$$

(9)

Where $f(x,y)$ is original image of size $MxN$ and $\hat{f}(x,y)$ is the reconstructed image.

We feel that our new performance parameter AFCPV gives better understanding of the closeness of the image as it is related to the perception. It takes into consideration the average fractional change in each pixel value. To void division by zero problem we have replaces zero by one in the denominator in equation (9). However it has been observed that there were not more than ten pixels having zero value in the original image.

Table I shows the time needed for encoding Bunny and Dragon images for codebooks of sizes 256 and 1024. The distortion of the encoded image is also shown for the codebooks 256 and 1024. The distortion of the encoded image remains same for all the algorithms since they are full-search equivalent.

Table II shows the search efficiency in the form of ratio evaluated by how many times the Euclidean distance computation is averaged performed compared to the size of the codebook. Search space for ES is considered as 100% and the reduced search space for other algorithms are compared to ES. A smaller ratio is better.

Table III Shows PSNR, Time, search efficiency and AFCPV for the proposed algorithm for different codebooks sizes.

Fig. 2 shows the results for Bunny image using codebook of sizes 256, 512, 1024 and 2048 encoded using proposed search algorithm.
TABLE III
RESULT OF THE PROPOSED ALGORITHM FOR DIFFERENT CODEBOOK SIZES

<table>
<thead>
<tr>
<th>Images</th>
<th>CB Size</th>
<th>PSNR</th>
<th>Time (s)</th>
<th>Search Efficiency</th>
<th>AFCPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunny</td>
<td>256</td>
<td>49.9</td>
<td>0.0041</td>
<td>0.10</td>
<td>0.6483</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>52.3</td>
<td>0.0042</td>
<td>0.05</td>
<td>0.485</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>54.2</td>
<td>0.0041</td>
<td>0.02</td>
<td>0.345</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>55.9</td>
<td>0.004</td>
<td>0.01</td>
<td>0.2488</td>
</tr>
<tr>
<td>Dragon</td>
<td>256</td>
<td>49.9</td>
<td>0.0042</td>
<td>0.10</td>
<td>0.4901</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>51.8</td>
<td>0.0041</td>
<td>0.05</td>
<td>0.3679</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>53.5</td>
<td>0.0042</td>
<td>0.02</td>
<td>0.2765</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>55.0</td>
<td>0.0043</td>
<td>0.01</td>
<td>0.2372</td>
</tr>
</tbody>
</table>

V. CONCLUSION

From Table I and Table II it is observed that proposed algorithm is faster as compared to other search algorithms since it requires considerably less number of Euclidean distance computations. Table III gives the performance of our proposed algorithm for different codebook sizes. The search efficiency in the form of ratio evaluated by how many times the Euclidean distance computation is averagely performed compared to the size of the codebook. Search space for ES is considered as 100% and the reduced search space for other algorithms is compared to ES. A smaller ratio is better. It is observed that proposed algorithm gives better search efficiency as compared to other algorithms. The newly introduced performance parameter AFCPV is computed for the proposed algorithm for different codebook sizes it is observed that larger codebook size gives lower value of AFCPV representing better human perception.

REFERENCES


Fig. 2 Results of Bunny image using proposed search algorithm for the codebooks of different sizes 256, 512, 1024 and 2048.


